Getting Acquainted with Unix and C: Design

Lucais Sanderson

January 23, 2023

1 Description of Program

At the top level, running the shell script, `plot.sh`, takes data generated from `monte_carlo` and generates several graphs in PDF form in the local directory. `plot.sh` accomplishes this by utilizing `gnuplot` to plot the respective graphs.

2 Files to be included in "asgn1":

1. `plot.sh`
    - Bash script that executes two `gnuplot` commands, redirecting respective `.plot` files to them.
2. `circle.plot`
    - Set of commands to be directed into `gnuplot`. Generates circle and sets the range that'll be plotted, `x:[0:1]; y:[0:1]`. Then `monte_carlo` is called with 2,000 iterations, piping the output to `grep`, the purpose being to exclude the first line containing column headers. All this is redirected into the `plot` command which plots the points, those in the circle being blue, otherwise they're red.
3. `error.plot`
    - Set of commands to be directed into `gnuplot`. Uses same method of redirecting `monte_carlo` output to `plot`, but uses `awk` as well to extract iterations and approximation columns. The difference between the approximation and π is what ends up being plotted so we end up with a Error vs Iteration graph. Functionality described more later.
4. `monte_carlo.c`
    - Source file that contains `main()` and produces the data table approximating π.
5. `riemann.c`
    - A tertiary C file that uses Calculus to derive π instead of `monte_carlo`.
6. `Makefile`

- Formats `monte_carlo.c` into clang-format and compiles it into binary `monte_carlo` with make from Makefile.

7. `README.md`

   - Text file in Markdown format that describes how to build and run the program, how the program handles erroneous inputs, and any problems encountered while developing the program.

8. `WRITEUP.pdf`

   - Shows example plots generated from the program as well as in-depth specifics on which UNIX commands were used and why.

9. `DESIGN.pdf`

   - Describes the design and design process of the program. Uses pseudocode. *This* document.

3 Pseudocode / Structure:

Note: Although I used redirection of files to make the bash script more readable, I'll describe it as though I used here-docs so the pseudocode is more readable here.

```
Tell gnuplot to output PDF format
Set the x and y range: [0:1], [0:1], respectively
Add circle
Define the color of the dot based on if it is in the circle or not
Plot, passing in shell expression that executes monte_carlo,
parsing it through grep

Tell gnuplot to output PDF
Set the y range to [-1:1] and x-axis to logarithmic scale
for 4 iterations:
    run monte_carlo with 20,000 iterations
    parse through awk: provide iteration number, PI - approx.
    plot
    wait 1 second
```

4 Note on `error.plot`

`error.plot` plots 4 different lines on the same graph. This is accomplished with an internal `for` loop inside the `plot` function. Due to how `monte_carlo.c` generates its seeds (generates based on time), I added a command in the input sequence being directed into `plot` that holds the program for one second. This gives enough time that the random seed generated is noticeably different and then that set of data is printed on the next iteration.

5 Credit

I take no credit for `riemann.c` at all. I derived `riemann.c` entirely from Andrew Oliver at:
towardsdatascience.com

Derived "grep -v 'x' " from user codaddict on stackoverflow at:
stackoverflow

Derived "set object circle ..." and "set size ratio -1" from users anyras and mgilson respectively at:
stackoverflow